Documentación del Proyecto

Aplicación Móvil (React Native)

Índice

1. Introducción del Proyecto

El propósito de esta aplicación móvil es ofrecer una red social de intercambio de imágenes similar a Instagram. Los usuarios pueden registrarse, iniciar sesión, interactuar con un feed de imágenes, dar "me gusta", realizar comentarios, editar su perfil y subir imágenes desde la cámara o la galería del dispositivo. Este proyecto utiliza React Native para Android y se enfoca en poner en práctica conceptos de frontend y seguridad mediante autenticación JWT.

2. Alcance

- Solo los usuarios autenticados tienen acceso a la aplicación.
- Incluye funcionalidad de interacción social (me gusta y comentarios).
- Actualmente enfocado en dispositivos Android; pendiente mejorar responsividad y soporte para iOS.

3. Arquitectura

3.1. Estructura del Proyecto

La estructura del proyecto se diseñó para ser modular, desacoplada y fácil de mantener:

```
socialapp/
|-- assets/         # Recursos est ticos como im genes
|-- components/     # Componentes reutilizables como PostCard y
   CommentSection
|-- context/        # Manejo de estado global (AuthContext,
   PostContext, ProfileContext)
|-- controllers/    # L gica de negocio y comunicaci n con la API
   (ej: postController)
|-- navigation/     # Configuraci n de navegaci n entre pantallas
|-- screens/        # Pantallas principales de la aplicaci n (Feed,
    Login, Profile, etc.)
|-- App.js          # Punto de entrada de la aplicaci n
|-- IP.js           # Configuraci n de la direcci n IP del backend
`-- package.json    # Configuraci n del proyecto y dependencias
```

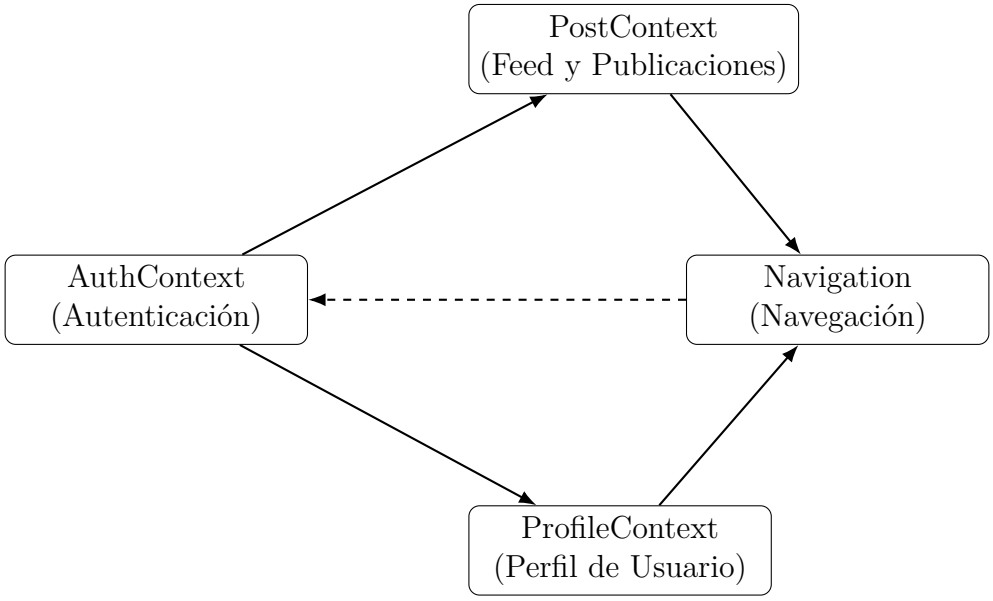
3.2. Justificación

- **Modularidad**: La separación por carpetas permite mantener un código organizado y facilita la búsqueda de errores.
- **Escalabilidad**: Esta estructura facilita agregar nuevas funcionalidades o adaptarlas a otros proyectos.

- **Reutilización**: Los componentes genéricos (como `PostCard` y `CommentSection`) pueden adaptarse a otros contextos.

3.3. Diagrama

Este diagrama ilustra cómo interactúan los diferentes módulos principales:



4. Manejo de Estado

- **AuthContext**: Centraliza la lógica de autenticación y autorización, maneja las credenciales del usuario y persiste su estado con `AsyncStorage`.
- **PostContext**: Administra el feed de publicaciones, la lógica de interacción (likes y comentarios) y la subida de imágenes.
- **ProfileContext**: Gestiona la información del perfil del usuario, incluyendo la actualización del nombre y la foto de perfil.

5. Patrones de Diseño y Buenas Prácticas

5.1. Patrones utilizados

- **CMV (Controller-Model-View)**: Separación clara entre lógica de negocio (*controllers*), datos (*contexts*) y vistas (*screens*).
- **SRP (Single Responsibility Principle)**: Cada controlador maneja exclusivamente una tarea, como autenticación o manejo de publicaciones.

Mejora futura: Implementar diseño atómico para mayor reutilización de componentes y facilitar actualizaciones.

5.2. Convenciones

- Nombres claros y descriptivos para variables, componentes y carpetas.
- Código modular y desacoplado, fácil de leer y mantener.

5.3. Componentes reutilizables

- `PostCard`: Maneja la visualización de publicaciones individuales.
- `CommentSection`: Permite agregar y mostrar comentarios en diferentes pantallas.

6. Integración de Tecnologías

6.1. Librerías clave

- **expo-image-picker**: Elegida por su facilidad de integración para acceder a la cámara y la galería.
- **@react-navigation**: Facilita la navegación entre pantallas, siendo intuitiva y bien documentada.

6.2. Tecnologías adicionales

- **AsyncStorage**: Para persistir datos del usuario y el estado global de la app.
- **Axios**: Para manejar las solicitudes HTTP con mayor flexibilidad.

7. Responsividad y Diseño

7.1. Estrategias actuales

Se implementaron estilos directamente en los componentes, lo que permitió un diseño funcional para dispositivos Android de tamaño estándar.

7.2. Limitaciones

Pendiente optimizar el diseño para que sea completamente responsivo y se adapte a dispositivos iOS y pantallas de diferentes tamaños.

7.3. Recomendaciones futuras

- Usar librerías como `react-native-responsive-dimensions` para manejar estilos adaptativos.
- Implementar un sistema de diseño centralizado con un tema global para facilitar cambios de estilos.

8. Dificultades Encontradas

8.1. Manejo de imágenes desde la cámara

- Retos técnicos al configurar permisos para acceso a la cámara y galería.
- **Solución**: Uso de `expo-image-picker` y revisión exhaustiva de permisos en dispositivos.

8.2. Manejo de estado global

- Sincronizar múltiples contextos (`AuthContext`, `PostContext`, `ProfileContext`) sin que interfieran entre sí.
- **Solución**: Modularización clara y uso de `useContext` para mantener la separación de responsabilidades.

8.3. Conexión con Expo Go

- Problemas intermitentes de conectividad durante pruebas en dispositivos físicos.
- **Solución**: Revisión de redes locales y pruebas constantes.

9. Lecciones Aprendidas

- **JavaScript vs TypeScript**: Aunque JavaScript es más rápido para implementar, TypeScript hubiera facilitado el mantenimiento a largo plazo gracias a su tipado estático.
- **Pruebas Incrementales**: Es crucial probar funcionalidades de manera incremental para identificar errores antes de avanzar.
- **Diseño Modular**: Una arquitectura desacoplada mejora la legibilidad y escalabilidad del código.

10. Conclusiones y Recomendaciones

10.1. Logros

- Se completaron todas las funcionalidades principales: autenticación, feed de publicaciones, interacciones sociales (likes y comentarios), y edición de perfil.
- La interfaz es funcional y cumple con los objetivos iniciales.

10.2. Aspectos a Mejorar

- Hacer la aplicación completamente responsiva y adaptable a iOS.
- Implementar un sistema de diseño centralizado (colores, tipografías).
- Adoptar TypeScript para mejorar la robustez del código.

10.3. Recomendaciones para futuros proyectos

- Planificar mejor la arquitectura y probar componentes desde el inicio.
- Adoptar principios de diseño atómico para una mayor reutilización y mantenimiento.
- Dedicar tiempo a optimizar la experiencia del usuario en pantallas pequeñas y múltiples plataformas.